

Building an API with Apigility

Rob Allen, October 2014

Apigility in action

Exercise 1: Build a ping API

Starting with the code in /exercise1, create an RPC ping service. Prove that it works using curl

Things to note:

- Turn on development mode
- Run the built-in webserver:

```
php -S 0.0.0.0:8080 public/index.php
```

Obstacles to a good API

API considerations

- Content negotiation
- HTTP method negotiation
- Error reporting
- Versioning
- Discovery

Other considerations

- Validation
- Authentication
- Authorisation
- Documentation



An opinionated API builder

<http://apigility.org>

JSON

Hypermedia Application Language (HAL) -
application/hal+json

```
{
  "_links": {
    "self": {
      "href": "http://localhost:8080/albums/1"
    }
  },
  "artist": "Eninem",
  "id": "1",
  "title": "The Marshall Mathers LP 2"
}
```

Error Reporting

API Problem - application/problem+json

```
{  
  "type": "/api/problems/forbidden",  
  "title": "Forbidden",  
  "detail": "Your API key is missing or invalid.",  
  "status": 403,  
  "authenticationUrl": "/api/oauth"  
}
```

HTTP Method Negotiation

POST /albums HTTP/1.1
Content-Type: application/json

405 Method Not Allowed
Allow: GET

OPTIONS

OPTIONS /albums HTTP/1.1

Content-Type: application/json

200 OK

Allow: GET

Accept

```
GET /albums/1 HTTP/1.1  
Accept: application/xml
```

```
406 Not acceptable  
Content-Type: application/problem+json
```

```
{  
  "type": "/api/problems/content",  
  "title": "Not acceptable",  
  "detail": "This API can deliver  
    application/vnd.music.v1+json, application/hal+json,  
    or application/json only.",  
  "status": 406  
}
```

Content-Type

POST /albums HTTP/1.1

Content-Type: application/xml

415 Unsupported Media Type

Content-Type: application/problem+json

```
{
  "type": "/api/problems/content",
  "title": "Unsupported Media Type",
  "detail": "This API can accept
    application/vnd.music.v1+json, application/hal+json,
    or application/json only.",
  "status": 415
}
```

Versioning by default

Media type:

`GET /albums HTTP/1.1`

`Accept: application/vnd.music.v1+json`

URL-based:

`/v1/albums`

Validation

```
PATCH /albums/1 HTTP/1.1  
Content-Type: application/json
```

```
{ "title": "" }
```

```
422 Unprocessable Entity  
Content-Type: application/problem+json  
{  
  "type": "w3.org/Protocols/rfc2616/rfc2616-sec10.html",  
  "title": "Unprocessable Entity",  
  "detail": "Failed validation",  
  "status": 422,  
  "validation_messages": {  
    "title": "Invalid title; must be a non-empty string"  
  }  
}
```

Authentication

- HTTP Basic and Digest (for internal APIs)
- OAuth2 (for public APIs)
- Event-driven, to accommodate anything else
- Return a problem response early if invalid credentials are provided

Authentication

```
GET /albums/1 HTTP/1.1
Authorisation: Basic foobar
Accept: application/json
```

```
401 Unauthorized
Content-Type: application/problem+json
```

```
{
  "type": "w3.org/Protocols/rfc2616/rfc2616-sec10.html",
  "title": "Unauthorized",
  "detail": "Unauthorized",
  "status": 401
}
```

Authorisation

```
GET /albums/1 HTTP/1.1  
Accept: application/json
```

```
403 Forbidden  
Content-Type: application/problem+json
```

```
{  
  "type": "w3.org/Protocols/rfc2616/rfc2616-sec10.html",  
  "title": "Forbidden",  
  "detail": "Forbidden",  
  "status": 403  
}
```

Hyperlinking: Pagination

Automatic when you return
`Zend\Paginator\Paginator`.

```
{
  _links: {
    self: { href: "/api/albums?page=3" },
    first: { href: "/api/albums" },
    last: { href: "/api/albums?page=14" },
    prev: { href: "/api/albums?page=2" },
    next: { href: "/api/albums?page=4" }
  }
}
```

Documentation

- Written within admin while setting up API
- Automatically populated via validation admin
- User documentation:
 - `apigility/documentation/{API name}/V1`
 - JSON or HTML based on accept header
 - Swagger available too

Use what you want

Write your own code, however ZF2 is under the hood.

Extend via...

- event listeners
- services

Let's talk about today's
application

Bookshelf application

- We have a collection of books.
- Users can borrow books.
- A logged in user can view their borrowed books.

Note:

- Authorisation and access control required
- Your clients require documentation!

The domain: Bibliotheque

A separate module, independent from the
Apigility code has our entities & mappers

Creating a REST service

Exercise 2: Books

Starting with the code in `/exercise2`, add an API called Bookshelf containing a REST service on the endpoint `/books` that can list all books & a single book.

Bonus points for creating, updating & deleting a book.

Things to note:

- Don't forget that all the domain code is in the separate module called `Bibliothèque`.
- `exercise2/README.md` is helpful

A quick look at exercise 2

Validation, filtering and documentation

Exercise 3: Validation

Starting with the code in `/exercise3`, add fields to the Users and Books REST services.

Bonus points for documenting them too!

Things to note:

- Validation rules in `exercise3/README.md`

A look at exercise 3

Authentication

Exercise 4: Authentication

Starting with the code in `/exercise4`, add OAuth2 to Users and allow logging in. Do not allow access to `/books` without a valid token

For bonus points, add a new endpoint `/books/borrowed`, that lists just that user's borrowed books.

Things to note:

- The database is set up to support OAuth2.
- A successful log in gives back a token for use with the Authorized header.
- More notes in `exercise4/README.md`.

A quick look at exercise 4

To sum up

- APIs provide many details to lose yourself in
- Apigility makes it easier

Thank you!

<https://joind.in/11761>

Rob Allen - <http://akrabat.com> - @akrabat

Resources

- <http://apigility.org>
- <https://github.com/zfcampus>

Lists & groups:

- <http://bit.ly/apigility-users> google group for support
- <http://bit.ly/apigility-dev> google group for dev discussions

Freenode:

- #apigility for support
- #apigility-dev for development discussion

Thank you!

<https://joind.in/11761>

Rob Allen - <http://akrabat.com> - @akrabat

Resources

- <http://apigility.org>
- <https://github.com/zfcampus>

Lists & groups:

- <http://bit.ly/apigility-users> google group for support
- <http://bit.ly/apigility-dev> google group for dev discussions

Freenode:

- #apigility for support
- #apigility-dev for development discussion